ELSEVIER

Computers in Industry 49 (2002) 283–298

COMPUTERS IN INDUSTRY

www.elsevier.com/locate/compind

An integrated approach for an interoperable industrial networking architecture consisting of heterogeneous fieldbuses

L. Hadellis[a,*], S. Koubias[a,1], Vassilios Makios[b,2]

[a]Applied Electronics Laboratory, Electrical and Computer Engineering Department, University of Patras, Patras 26500, Greece
[b]Electromagnetics Laboratory, Electrical and Computer Engineering Department, University of Patras, Patras 26500, Greece

Received 7 June 2001; accepted 31 May 2002

Abstract

An integrated approach for an interoperable networking architecture is proposed and implemented within a distributed heterogeneous industrial enterprise environment consisting of two fieldbus segments, a LonWorks and a Profibus, interoperating through an Interoperability Unit (IU) developed and built on a Windows 2000 platform. The proposed approach introduces an interoperability layer above the OSI stack extending the ISA/SP50 User Layer philosophy that contains a model of common objects and messages for automation applications and is called interoperable domain. The real automation distributed application is mapped to a Virtual Application (VA) that is introduced within the interoperable domain and links the distributed parts. Messaging mechanism is based on supplier common object method requests by client common objects along a relationship within the virtual application. A data-oriented format is employed for messages avoiding explicit event transfer. Finally a framework for the standardization of automation functions and services is proposed.

Keywords: Interoperable; Networking architecture; Fieldbuses; Industrial communication

1. Introduction

The requirement for a global interoperability in a heterogeneous industrial enterprise environment consisting of different fieldbuses and data networks is recently a field of continuous research that is still far from being satisfied. Modern industrial enterprises contain formal, global, functionally partitioned and modular systems as far as automation, control and communications are concerned. At the same time, the whole industrial enterprise structure should be flexible, distributed, integrated and interoperable.

Generally, interoperability is achieved when heterogeneous operating entities can communicate transparently and work together for a common scope. These individual entities, in order to interoperate, should in some way represent their internal living structure to a corresponding external common and generally understandable behavior. The translation to the common behavior consists of a "think in common" aspect as well as a "talk in common" aspect. The first aspect refers to a new common language, which implies the introduction of new common objects. The second aspect refers to an agreed upon behavior based on common interoperable messages that normally act as

* Corresponding author. Tel.: +30-610-369258; fax: +30-610-369298.
E-mail addresses: loukas@teipat.gr (L. Hadellis), koubias@ee.upatras.gr (S. Koubias), v.makios@ee.upatras.gr (V. Makios).
[1] Tel.: +30-610-997299; fax: +30-610-997333.
[2] Tel.: +30-610-997286.

behavior selectors. Thus, heterogeneous entities with different internal structures can transparently interact within a common interoperable domain by means of a behavioral shell, as presented by Hadellis and Koubias [2], that makes them all adapt within the domain instead of converting each time to a specific different format in order to work together with some other entity.

In this work, an integrated approach with application is proposed and implemented employing two heterogeneous fieldbus segments, a LonWorks and a Profibus, interoperating and supporting a distributed automation application through a proposed Interoperability Unit (IU) developed and built on Windows 2000 platform. The high-level interoperability policy of the proposed approach is based on the introduction of an interoperable domain where real heterogeneous elements and the distributed application are mapped. Then, moving downwards, specific data-driven interoperability techniques and mechanisms based on a certain platform are being proposed, developed and implemented. However, existing technology issues, fieldbus specific characteristics, current models, standards and other factors bring feedback from below and introduce upward approach paths at specific design phases. The proposed interoperability framework in the distributed heterogeneous environment is approached and developed in layers, starting with the introduction of an interoperability layer, above the OSI 7-layer stack, as it appears in Fig. 1. This layer extends the philosophy of the ISA/SP50 [14] User Layer from its homogeneous fieldbus framework supporting multivendor device interoperability, to a heterogeneous fieldbus framework supporting distributed applications. It implements the interoperable domain behavioral shell above each heterogeneous fieldbus, within the industrial enterprise environment and forms a basis for higher-level enterprise objects and services to be developed towards an end-to-end application level interoperability. Low-level details and complexity tend to be hidden behind higher-level standardized functions and services.

The interoperability layer is designed to be independent from the communication infrastructure. The underlying communication network infrastructure and associated services will not affect interoperability functionalities but they will affect timeliness and reliability. Real time operating platforms, certain LAN and WAN real time protocol versions and switching technologies, may support the real time operation of the interoperability mechanism, concerning the requirements that could satisfy the predictability and controllability of the network delays.

The structure of this paper is as follows: Section 2 gives the current status concerning interoperability issues in industry. In Section 3, a common object model framework for automation applications is proposed. Then in Section 4, the interoperability mechanism is presented along with the IU architecture and an indicative application. Finally, Section 5 contains the conclusions.

2. Current status

Today, many different fieldbus concepts and practices exist and continue to be developed in industrial and building automation. The large number of different device and fieldbus types within a control system makes the integration and engineering tasks difficult. The idea to create a universal fieldbus system is

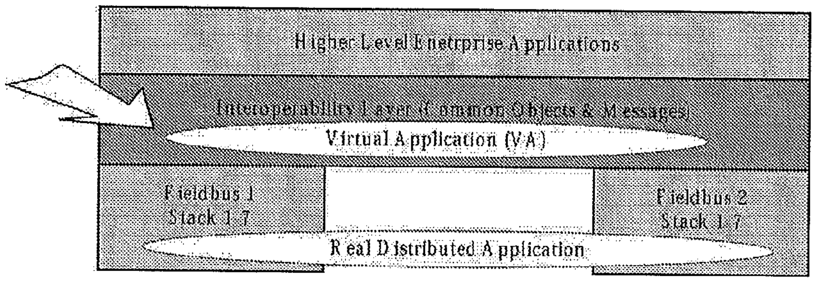

Higher Level Enetrprise Applications
Interoperability Layer (Common Objects & Messages)
Virtual Application (VA)
Fieldbus 1 Stack 1-7
Fieldbus 2 Stack 1-7
Real Distributed Application

Fig. 1. The proposed interoperability layer.

examined but still is not applicable due to many reasons, as also mentioned by Dietrich and Sauter [3]. It seems that real industrial world should accept its polymorphism and proceed to develop methods to integrate these heterogeneous parts under higher-level entities.

Evolving standards such as the IEC 61499 [5] and IEC 61804 [6] define the basic concepts and a methodology for the design of modular, reusable, distributed components for industrial control systems. They define the function block construct as the main building block of Industrial Process Measurement and Control Systems (IPMCSs) applications. The function block encapsulates industrial algorithms with inputs and outputs in a generic form that can be effectively understood and applied. Complete applications can be built from interconnected function blocks that can be spatially distributed in many intelligent devices.

Several approaches employed the function block concept to propose and develop architectures that support interoperability in general. The Profinet approach [9] develops an environment employing a COM object framework that may support interoperability on an Ethernet-TCP/IP basis integrating for the moment only Profibus systems and claiming to be open for other fieldbus systems through proxies. Profibus [7,8] employs function blocks to build standard functional modules in order to support and develop distributed applications within a homogeneous Profibus environment. LonWorks similarly employs the function block concept in LonPoint/LonMaker system [12] providing a variety of standard software function blocks. LonPoint function blocks are LonMark objects [10,11] and are employed to support interoperability within the homogeneous LonWorks environment as defined by the LonMark Interoperability Association. Both approaches support interoperability among different vendors within a homogeneous fieldbus environment.

Fieldbus standard ISA/SP50 [14] User Layer located above the typical ISO/OSI 7-layer stack introduces an eighth layer that is designed specifically to support process control applications. Fieldbus Foundation employs this standard along with function blocks and device descriptions to develop and propose a single international, interoperable new fieldbus standard. These efforts concentrate on a new fieldbus standard development supporting multi-vendor device interoperability rather than integrating existing heterogeneous fieldbus technologies thus supporting distributed applications interoperability in a heterogeneous fieldbus environment.

Extensive work by Marcos et al. [15] and by Marti et al. [16] is being carried out concerning remote monitoring, supervision and control of fieldbus-based industrial processes with the use higher-level objects supported by Java, Corba, COM/DCOM and TCP/IP. New plant automation structures and concepts concerning remote monitoring and control of manufacturing processes, such as the virtual plant model, are being developed and can be used to build flexible production sites.

OPC [13] communication standard provides an efficient way that allows multiple software programs or applications, within the industrial control domain to talk to each other. It operates as a translator shell that reads data (OPC server) from specific fieldbuses and converts them to a neutral form understood by many software application platforms (OPC client). Therefore, it is employed as a structural element for interoperability development but since it cannot provide end-to-end interoperability alone, it requires additional software power when heterogeneous fieldbus distributed applications are concerned. OPC-DX [21] initiative is an OPC data access specification extension that defines an interface standard for interoperable data exchange and server-to-server communication across an Ethernet network and certain fieldbus systems that have agreed to support this initiative (Fieldbus Foundation HSE, ProfiNet/ProfiBus, ControlNet and DeviceNet).

Other research work by PABADIS [17] is also based on decentralized, independent, intelligent and autonomous objects representing physical and logical control and data elements employing the agent concept in Java platform. However this work is mainly focused on control intelligence distribution through software agents and new generation PLCs with JVM infrastructure, rather than integrating existing heterogeneous fieldbus dependent PLCs and controllers.

Finally, extensive work is being carried out concerning device description, specifications, methodology and tools by Thramboulidis and Tranoris [18], by Neuman et al. [19] and by Thramboulidis and Prayati [20] for engineering support systems employing the function block concept and UML methodology.

Most of the previously mentioned works refer to either homogeneous fieldbus environment device interoperability or to remote plant monitoring and control. OPC and OPC-DX deal with the heterogeneous fieldbus interoperability problem through the development of a data exchange and server-to-server communications standard across an Ethernet backbone domain. Interoperability among heterogeneous fieldbuses is a multi-dimensional problem requiring parallel development in all these fields such as device profiles, functional blocks, higher-level objects, new ISO/OSI layers, proxy techniques and standardization of services.

Our approach concentrates on the interoperability issue between heterogeneous fieldbuses without any backbone network prerequisite. It defines, develops and implements an interoperability environment based on certain concepts, higher-level principles and specific mechanisms implementing these concepts and principles. It introduces specific new elements and integrates them appropriately in a distributed control application framework consisting of the two popular fieldbus technologies, LonWorks and Profibus. These fieldbuses interoperate through an Interoperability Unit (IU) developed and implemented in a Pentium III 866 MHz Windows 2000 platform.

3. A common object model framework for automation applications

This approach introduces an interoperability layer that extends the ISA/SP50 User Layer philosophy and provides the basis for further higher-level interoperable functions, tasks and services, up to the enterprise level, within a heterogeneous fieldbus environment. This layer contains a common object model and the corresponding common messages. Our approach introduces a common object model with respect to LonWorks [4] and Profibus object models [8]. Analytically, we introduce eight (8) common object classes that include homogeneous automation objects. It should be pointed out that the proposed object classes catalog should be capable of continuously integrating any newly introduced instances or classes. These common object classes, as it is shown in Fig. 2, are the Actuator Class, Sensor Class, Control Algorithm Class, Node Class, Alarms Class, Log Class, User Interface Class and General Class. The actuator, sensor and control algorithm classes are closely related to device description standard concepts and to the function block concept respectively. All heterogeneous objects should map accordingly to this model and be considered instances of any of the above classes. Some of these classes may also be found to levels higher than the automation application layer such as the alarms, log and user interface class at the enterprise application layer.

Modern industrial enterprise control and automation systems tend to become increasingly vertically integrated. This introduces an additional application-layer categorization scheme for automation objects as well as higher-level objects. The structure, shown in

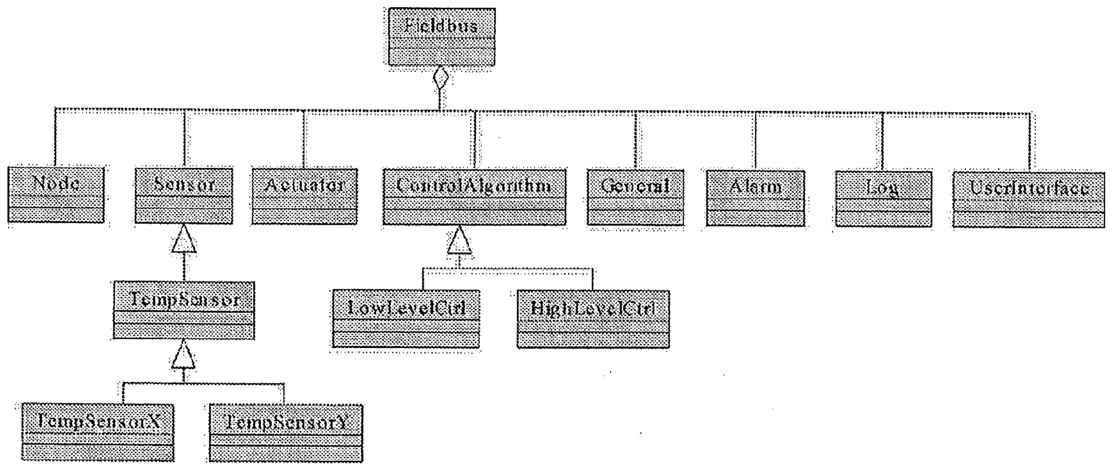


Fig. 2. Fieldbus components aggregation with inheritance samples.

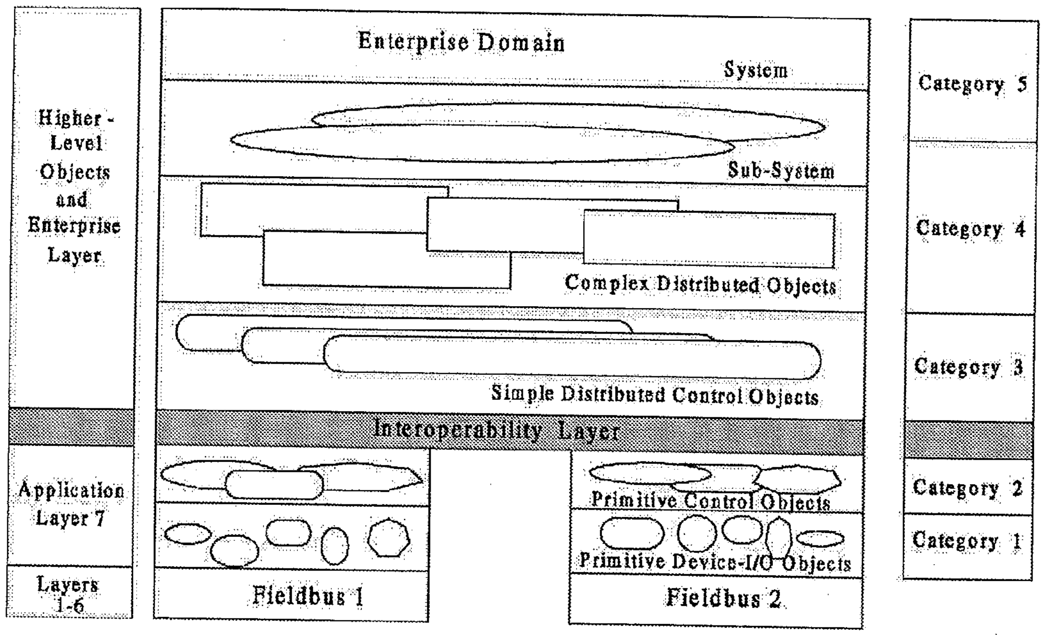

Higher-Level Objects and Enterprise Layer
Application Layer 7
Layers 1-6
Enterprise Domain
System
Sub-System
Complex Distributed Objects
Simple Distributed Control Objects
Interoperability Layer
Primitive Control Objects
Primitive Device-I/O Objects
Fieldbus 1
Fieldbus 2
Category 5
Category 4
Category 3
Category 2
Category 1

Fig. 3. An industrial enterprise vertical structure in terms of objects control levels and complexity.

Fig. 3, provides a more detailed description of the automation-level objects relative position across the enterprise total vertical structure. Object classes are presented with respect to the ISO-OSI Reference Model (RM). Five categories of object classes are considered.

(a) Category 1. It includes all low-level automation object classes such as sensors and actuators.
(b) Category 2. It includes the lower part of the control algorithm class that is internal to the control node but it may also include low-level alarm, data log, user interface elementary functions and the node class. It represents low-level controlling functions such as I/O read/write, regulation, etc. hidden behind higher-level functions and services and corresponds to the second application layer subdivision with respect to Figs. 5 and 6 of Chapter 4.1.
(c) Category 3. It includes the higher part of the control algorithm class and represents higher-level controlling distributed functions such as closed loops, etc. integrating low-level control elements. Interoperability starts at this layer. Higher-level alarms, logging and user interface functions may reside here supporting the plant area management.
(d) Category 4. It represents the sub-system and refers to the plant level consisting of interconnected heterogeneous fieldbuses at different areas. Higher-level services, that integrate category 3 functions, reside here including sub-system alarms, logging and graphical user interfaces.
(e) Category 5. It represents the whole system made of spatially dispersed plants across the enterprise. Enterprise level services, complex object models and applications reside here.

With respect to this view, the interoperability layer is located between categories 2 and 3 and it maps all category 1 and 2 objects that are fieldbus dependent, to the common model. Above it, a framework of layer 3 functions has to be introduced and developed. This will support layers 4 and 5 complex functions and services.

According to this approach, an object complexity factor appears and a complexity-oriented view may be introduced as it is seen in Fig. 3. The following types are proposed:

(a) Primitive device or I/O dependent local objects. They cannot be divided further to simpler entities (a digital input for example). They are local to a node being part of category 1.
(b) Primitive control local objects. They cannot be divided further to simpler entities (a comparison algorithm using variable values is an example). They are single process objects, local to node being part of category 2.
(c) Simple distributed control objects. They are considered as a mixture of primitive local objects that participate in a distributed application (a simple distributed control algorithm is an example). They are distributed within a network being part of category 3.
(d) Complex distributed objects. They are considered as a mixture of simple distributed objects and they actually represent high-level services and functions (a complex closed control loop). They are multi-process objects since for the control they support, they involve many processes being part of category 4.
(e) Sub-system objects. They are standard service composite objects offering high-level services (a heating sub-system is an example). They may also be distributed across LANs or WANs being part of category 4 or 5.
(f) System objects. They are domain composite objects or components at the enterprise level. They are widely distributed within LANs or WANs being part of category 5.

According to the above definitions, all devices and low-level control entities will be able to be standardized and categorized depending on their features. For example, a pressure sensor belongs to the sensor class, resides in the first category and is a primitive object. These features have to be standardized and the same standard has to be employed for all homogeneous devices independent of the vendor or manufacturer. Then, concerning the higher-level objects that offer higher-level functions and services a certain framework has to be developed with rules that may assist their categorization and standardization. An introduction of standard automation application functions and services is implied here. A very important lesson can be taught from the computer evolution. TCP/IP protocol is the interoperability layer concerning heterogeneous computer systems communication and further close cooperation. However, this was not enough. Application-level standard services were introduced above the TCP layer, such as http, ftp, odbc, ole, ppp, etc. in order to enhance interoperability. Following this paradigm, it is considered that the development of standard automation application functions and services is the next necessary step in order to obtain a powerful interoperability framework, that will effectively hide local devices and practices below a common objects enterprise and management environment. As far as the functions are concerned this task is being currently developed through the function block standardization efforts. However, the services have also to be standardized. LonMark has developed such service standardization within a LonWorks environment.

Therefore, a framework is proposed for this development. This framework may become the basis for independent but interoperable development of services and service object classes. A practical example from the industrial domain may be the following. Heterogeneous cooperating devices such as temperature sensors, e-valves, flow meters, boiler, pressure meters, etc. from heterogeneous fieldbuses can be effectively hidden behind a standardized heat service that consists of several functions such as the boiler control, the hot water flow control and the alarm/safety control functions. The device, the function and the service entities and concepts can be clearly seen here. An automation application service, such as the heat service in the paradigm, according to the proposed approach requires a set of specific common messages in order to be defined. This set of messages should be defined in order to be clearly related to the specific service. In this way, standard functions and services can be clearly defined with the use of corresponding sets and super-sets of common messages exchanged between common objects involved in the function or service.

Returning now to the proposed common object model, it should be pointed out that this model has to be "public" by means of an object directory. Therefore, a common object server entity has to be introduced that hosts all common objects. Heterogeneous fieldbuses will be the clients that are notified about the interoperable domain common objects and messages in order to use or act upon them.

4. The proposed Interoperability Unit (IU)

4.1. The interoperability mechanism

The architectural structure of the proposed interoperability mechanism is presented here. At first the high-level interoperability policy is based on the following three principles:

(a) A common interoperable object model must be created within the interoperability layer. This model realizes the common language.
(b) A behavioral shell must be developed, consisting of interoperable messages with agreed upon meanings. These messages are closely related to the common objects.
(c) A standardized framework containing automation functions and services must be proposed to hide all low-level control-type elements, such as devices and controlling units under higher-level service-type elements.

According to these principles the heterogeneous fieldbus local objects will converge to a semantically stable and unambiguous interoperable environment. The common language and the messages will be created in a formal way. Therefore, the common object and message format will be independent from its meaning, and the semantics and syntax will be based on a unique, clear and primary relation between the entity to be represented and its representation. Common messages operate as interoperable domain value carriers between common object instances representing the real local objects. Sets of messages define functions and super-sets of messages define services. The messaging structure is based on a data-oriented format. Since event transferring is instantiated through data transfer (i.e. an event may be defined through the change of a variable value), events are implicitly transferred through variable updates. This approach lightens the message repertoire, simplifies information exchange and keeps system complexity at a low-level as far as messages are concerned.

The basic structure and topology of the proposed heterogeneous system appears in Fig. 4 consisting of two control nodes located at two different fieldbuses interoperating through the IU. The process control application is distributed in both nodes and fieldbuses. An enterprise level application residing in a PC may be accessed via a WAN or LAN connection. The IU Unit incorporates appropriate interface ports in order to connect to the fieldbuses and the WAN.

The process control application scenario is based on a trans-node distributed closed-loop control application

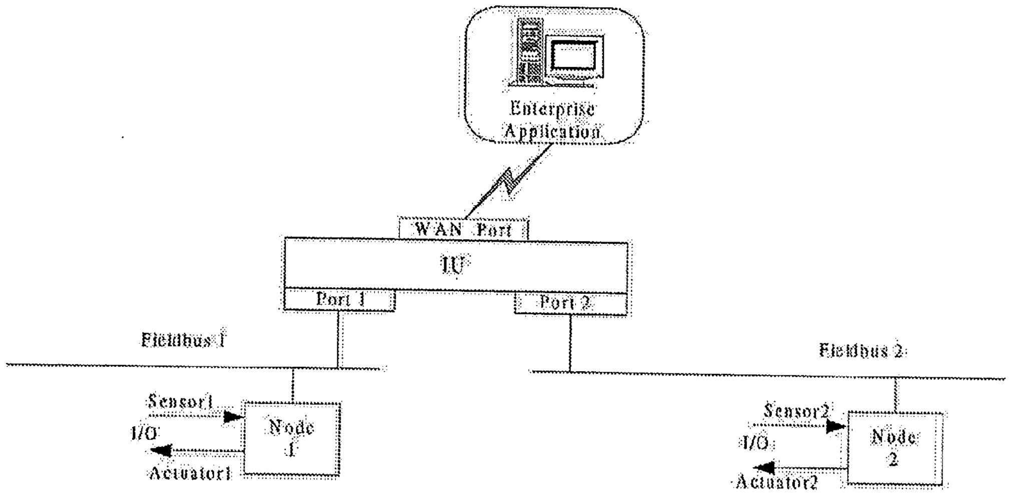


Fig. 4. The proposed structure and topology of the heterogeneous networking system.

involving event-driven and aperiodic message arrivals (from I/O or network). Data types exchanged are considered stochastic and critical.

The interoperability mechanism developed and implemented resides within the IU. Fig. 5 provides an insight to the interoperability mechanism while Fig. 6 gives an OSI-RM layered view of the approach. All node messages that are part of the distributed application enter the IU. These messages are then translated to the common language within the interoperability domain. The translator layer, according to Fig. 6 acts as a separate relay layer or wrapper layer that maps each individual fieldbus local object and message to the common objects and messages.

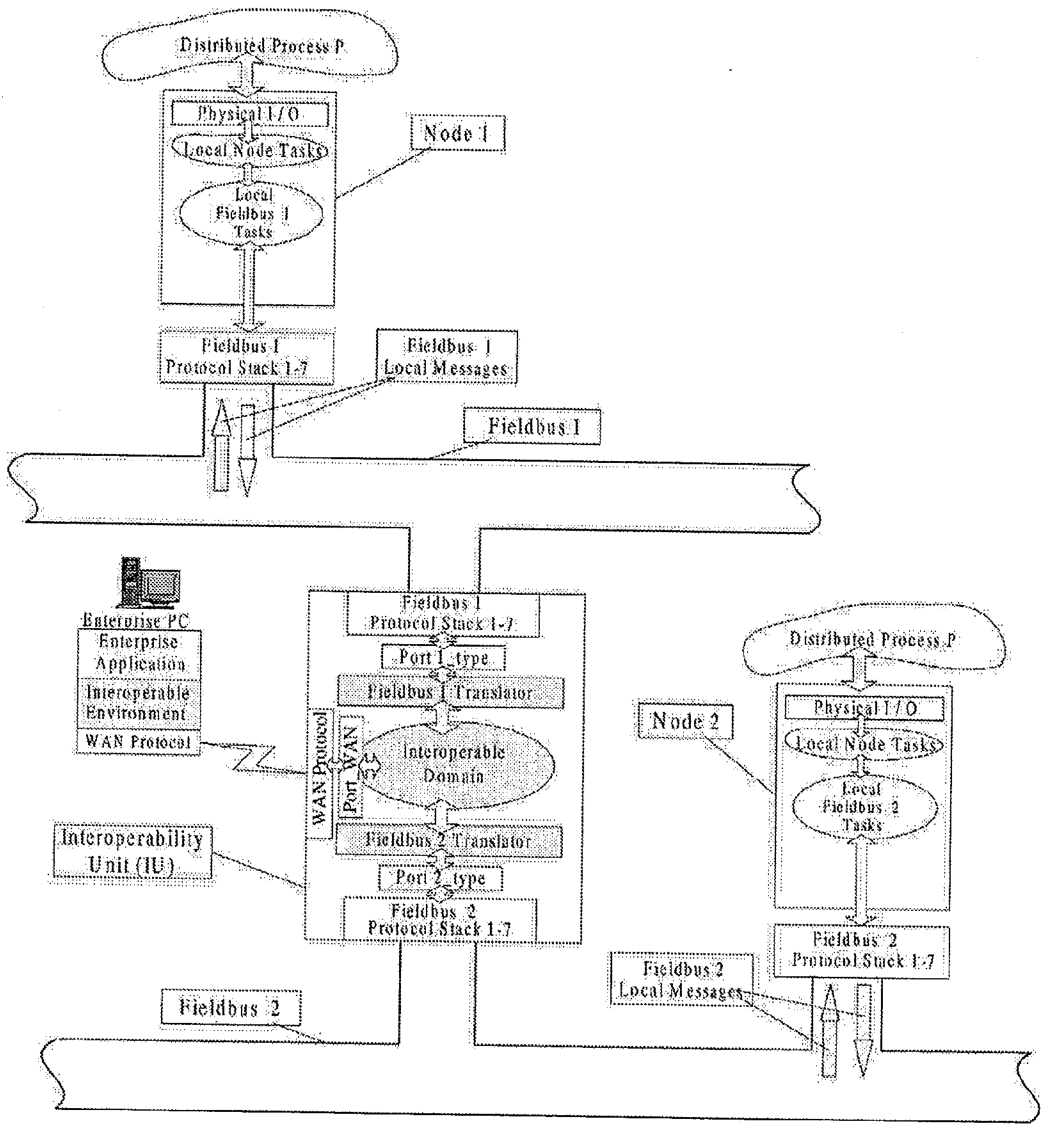


Fig. 5. The interoperability mechanism architectural structure.

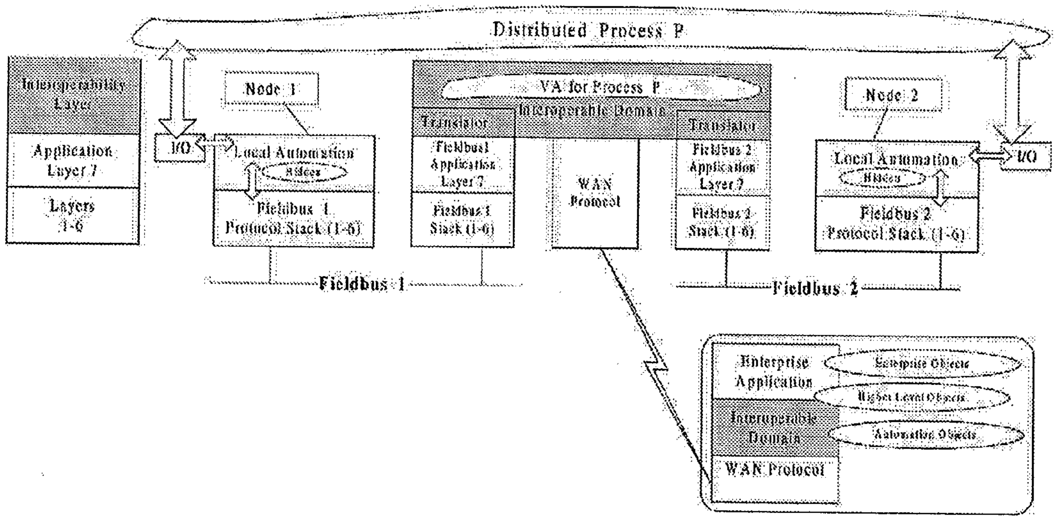


Fig. 6. An OSI-RM oriented layered view of the interoperability mechanism.

The real distributed application that controls the distributed process is mapped to a Virtual Application (VA) that exists and runs within the interoperable domain. The linking of the heterogeneous distributed application separate parts is being realized within the interoperable domain through the VA that consists of the common objects and messages.

For each node two application level subdivisions are considered, according to the ISO/OSI-RM, as it is shown in Fig. 6. The first subdivision corresponds to the node physical I/O that "touches" the industrial process. The second subdivision, corresponds to a local automation level that is node/fieldbus dependent, is internal and acts within the node and the fieldbus. The local automation level is further divided into a local node tasks part and a local fieldbus tasks part, as it shown more analytically in Fig. 5. It also includes the hidden local part of the distributed application.

All interoperable common messages will be produced and consumed within the IU unit. IU ports, specifically Port 1_type and Port 2_type, represent the specific fieldbus application layer interface to the interoperable domain translator. The enterprise application accesses directly the interoperable domain and the virtual application through Port_WAN and the WAN protocol without any translation.

It should be pointed out that there is a functional independence between the interoperable domain and the communication infrastructure and services. Communication affects only timing and further reliability when timing is poor. It is assumed that there is no interdependence between communication and interoperability mechanisms.

If the embedded agent concept is employed within the fieldbus nodes then the interoperability mechanism may be partially residing within the nodes and partially within the IU. The "Heavy Node" (HN) term is introduced that is a fieldbus node with an interoperability agent embedded implementing interoperability mechanisms at the node level. Three application level subdivisions can be considered then for each HN according to the ISO/OSI-RM. The first and second subdivisions are similar to the ones described earlier. The third subdivision that exists only within a HN corresponds to a part of the interoperability layer, which implements the node behavioral shell.

The local object and message translation to the common objects and messages is partially being carried out within the node. Therefore, HNs should contain two parallel messaging routes and corresponding ports, one for the interoperable distributed domain and one for the local fieldbus domain. At the end, both interoperable and local messages are encapsulated within the fieldbus stack in order to be transferred to the network.

This IU version can be considered as the light one, since interoperable messages are produced and consumed within heavy nodes. Only common interoperable messages can pass through the IU. However the limits between node-hosted interoperability mechanism part and the IU-hosted part requires further examination that is out of the scope of the present work.

4.2. *The IU architecture*

The IU generic and application specific architecture appears in Fig. 7. It was developed and implemented in a Windows 2000 Pentium III 866 MHz PC. Two fieldbus interfaces, a Profibus Card (Master) and a LonWorks SLTA-10 serial interface adaptor, connect the IU to the fieldbuses. OPC is employed as the translator between the fieldbus proprietary environment and the common objects, depersonalizing the fieldbus internal characteristics to a COM-based generic format. OPC operates as a neutral shell for each fieldbus performing a fieldbus wrapper task. Then the depersonalized items are given new names according to the new language that is realized by the common object model. The common objects and messages interoperable environment with the Virtual Application (VA) was developed and implemented in Visual Basic (VB) 6.0 supported by OPC Automation 2.0 from VB6.0 references.

The IU is proposed to be a multi-port modular device based on a real time operating system incorporating at least two or more fieldbus interface ports with the required fieldbus protocol stack embedded that can be added in available IU slots. Even if the interoperable domain at its first conceptual definition tends to be considered something external and spatially extended, it is essentially an internal programming environment within a workstation real time H/W and S/W platform.

The real time aspect is affected mainly by the real time capabilities of the IU workstation platform. The current approach is mainly focused on interoperability with the real time aspect requirement relaxed. However, Windows 2000 extensions as proposed by Rosen and Oberland [1] and OPC event driven operation could enhance the real time aspect. Furthermore, an embedded version of the proposed IU supported by an RTOS along with a custom OPC in C language can provide a true real time behavior.

The existence of an intermediate WAN or LAN connection between the IU and an enterprise PC, as it appears in Fig. 4, or in the case where the two fieldbuses are far apart and two IUs are employed connected through their WAN ports, additional latency is introduced yielding reduced reliability. In this approach, the communication networks infrastructure and transfer service is decoupled from the interoperability mechanism. The common object model and messages is functionally independent from

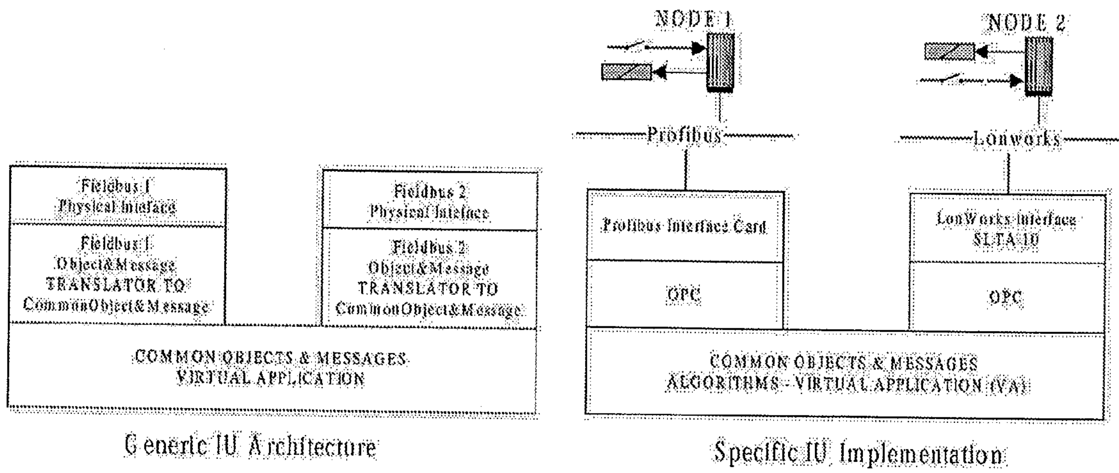

NODE 1
NODE 2
Profibus
Lonworks
Fieldbus 1 Physical Interface
Fieldbus 1 Object&Message TRANSLATOR TO CommonObject&Message
Fieldbus 2 Physical Interface
Fieldbus 2 Object&Message TRANSLATOR TO CommonObject&Message
COMMON OBJECTS & MESSAGES VIRTUAL APPLICATION
Profibus Interface Card
OPC
LonWorks interface SLTA 10
OPC
COMMON OBJECTS & MESSAGES ALGORITHMS - VIRTUAL APPLICATION (VA)
Generic IU Architecture
Specific IU Implementation

Fig. 7. The generic and specific IU architecture.

the communication infrastructure. However, there is a strong timing and reliability dependence due to network congestion, delays and lost packets. It is considered that the employment of switching technologies at the LAN or WAN level can positively improve congestion and minimize delays. The careful examination of the real time protocol (RTP) and reservation protocol (RSVP), along with the latest developments in real time services for data networks, may provide useful results and techniques that enhance performance and reduce latency to acceptable levels.

4.3. An indicative application

A real distributed application was designed and developed in order to realize the proposed integrated approach to interoperability. LonWorks and Profibus were selected as the two heterogeneous fieldbuses with one control node for each, as it appears in Fig. 8. The real application is a closed-loop digital control application employing input switches and output relays with feedback. Digital input switches, Sw1.1 and Sw1.2, located at a Profibus Siemens PLC node inputs I0.0 and I0.1, control the state of relay digital outputs, Rel2.1 and Rel2.2, at a LonWorks node outputs IO_1 and IO_2. The relay secondary contacts, RelCt2.1 of Rel2.1 and RelCt2.2 of Rel2.2, are fed to the LonWorks node digital inputs IO_4 and IO_5, that return a feedback about the relay states back to the Profibus Siemens PLC node relay digital outputs Rel1.1 and Rel1.2, through the Q0.0 and Q0.1 digital outputs.

All local heterogeneous objects, which are involved in this application, have to be mapped accordingly to the corresponding common objects. Four automation objects can be recognized here: the switch, the relay, the relay contact and the control algorithm. However, three common object classes are involved: the two-state switch sensor class (including the switch and the relay contact), the relay actuator class and the control algorithm class related to the relay control by the switch state. These classes are derived from the common object UML model of Fig. 2. Specifically, the sensor and the actuator common object classes appear in Fig. 9 with their properties and methods. They are named Com2StateSwitch and ComRelay accordingly.

The control algorithm specific class, named ComSwRelCtrlAlg, appears with its relationships to the other classes in Fig. 10 that shows the indicative application VA UML model. A sensor listener class, named ComSwitchListener, is also introduced there. The control algorithm class inherits the ComSwitchListener methods, which act as its inputs. Then acting as a client object the control algorithm class requests the ComRelay actuator object functionalities. ComRelay acts as a supplier object and offers its functionalities by supplying its methods. The common messages are mapped to the supplier class methods. Therefore, the common messages become the supplier common object class methods.

In this way, the real automation application is mapped to a virtual one (VA) within the interoperable domain that contains the common objects. Common object instances act as virtual representatives of real heterogeneous local objects involved in the process. The virtual application consists of the common objects involved and their relationships that correspond to the common messages. The set or super-set of common

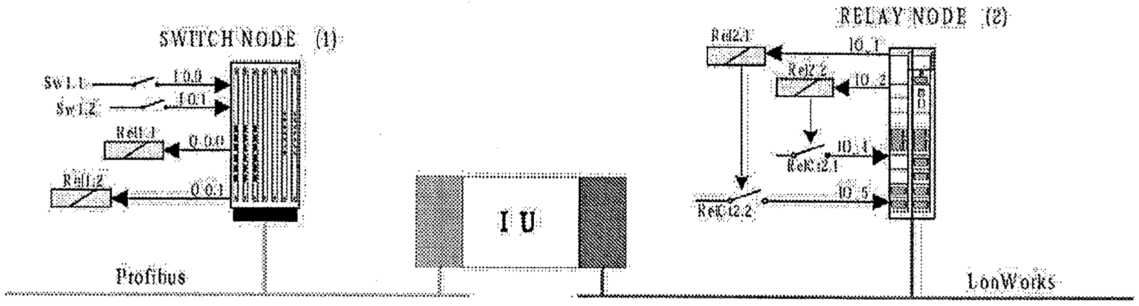


Fig. 8. The closed-loop digital control application hardware.

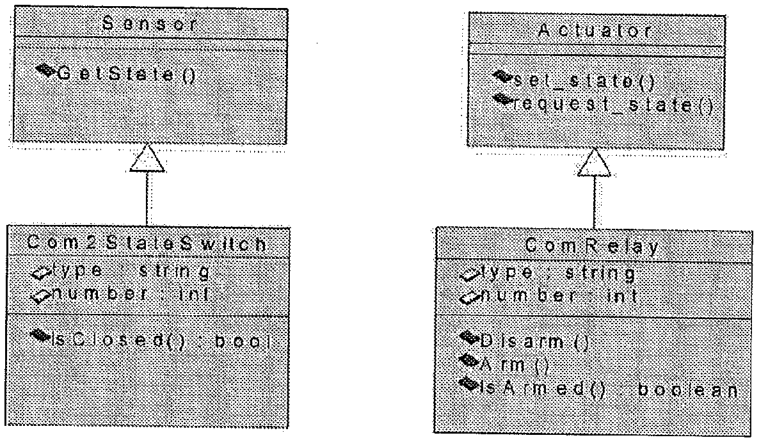


Fig. 9. The sensor and the actuator common object classes in UML.

messages created and consumed within the virtual application interoperable domain can be used to define a standard function or service thus supporting the standardization of functions and services.

Real objects are represented by common objects instances. Real sensor Sw1.1 is represented by the SimSw1.1 object (simulated Sw1.1) that belongs to Com2StateSwitch class. Real actuator Rel2.1 is represented by the SimRel2.1 object (simulated Rel2.1) that belongs to the ComRelay class. The control algorithm is represented by the SimSw1.1CTRLSimRel2.1 (simulated switch controls the state of simulated relay) object that belongs to the ComSwRelCtrlAlg common object class. The virtual application consists of the simulated objects involved along with their relationships that correspond to the common messages. Finally, the common messages are the supplier common object methods requested along a relationship. The virtual application has already been built, downloaded to the IU and running. This procedure appears

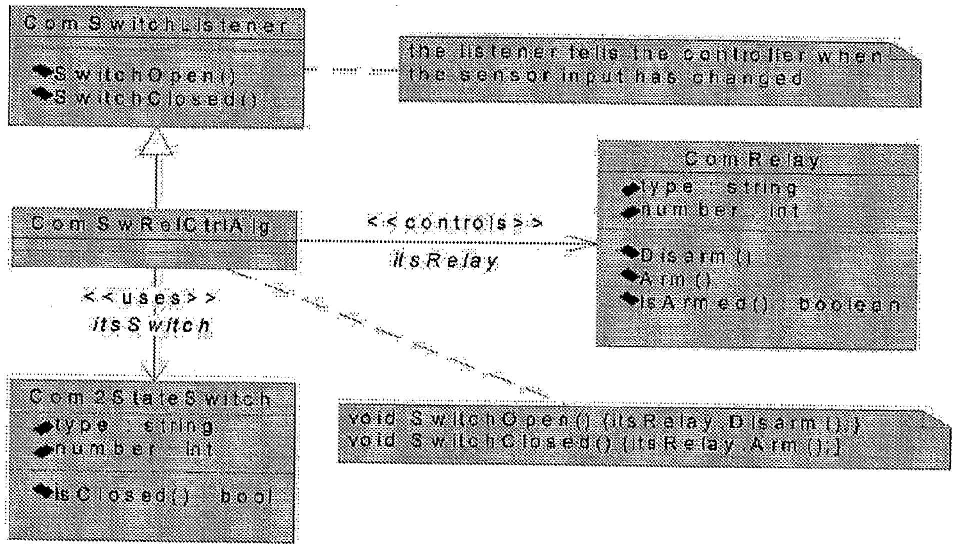


Fig. 10. The indicative application VA UML model.

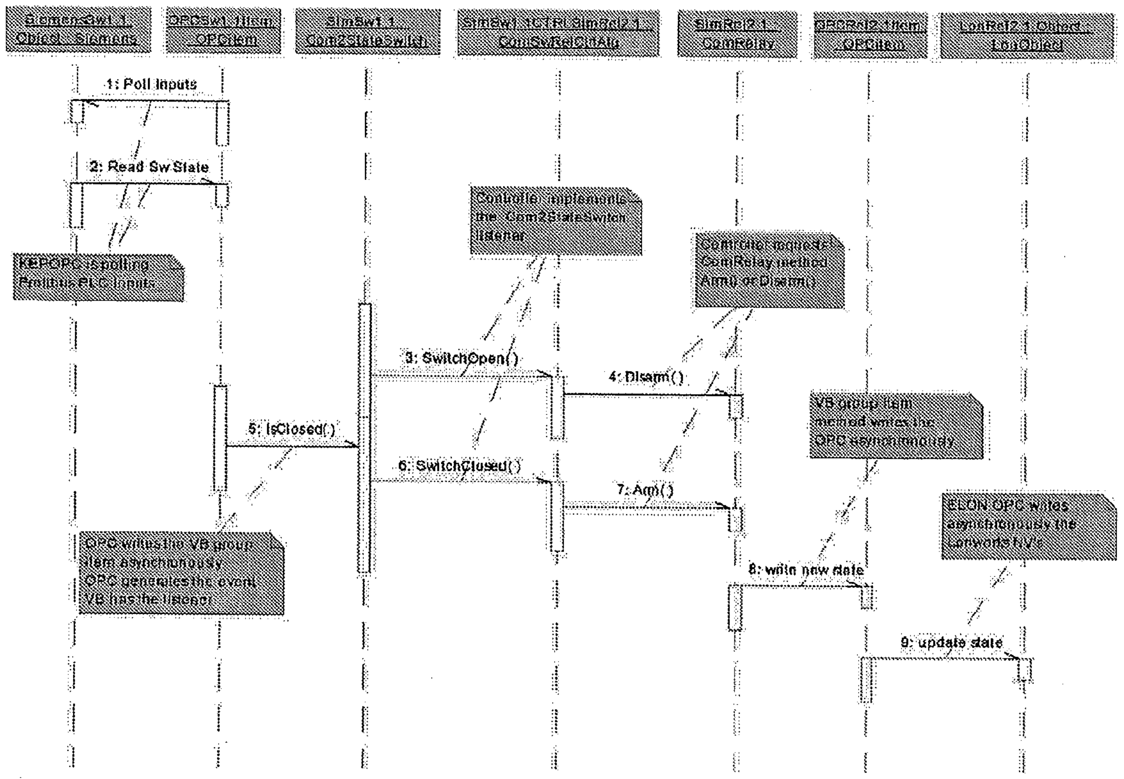
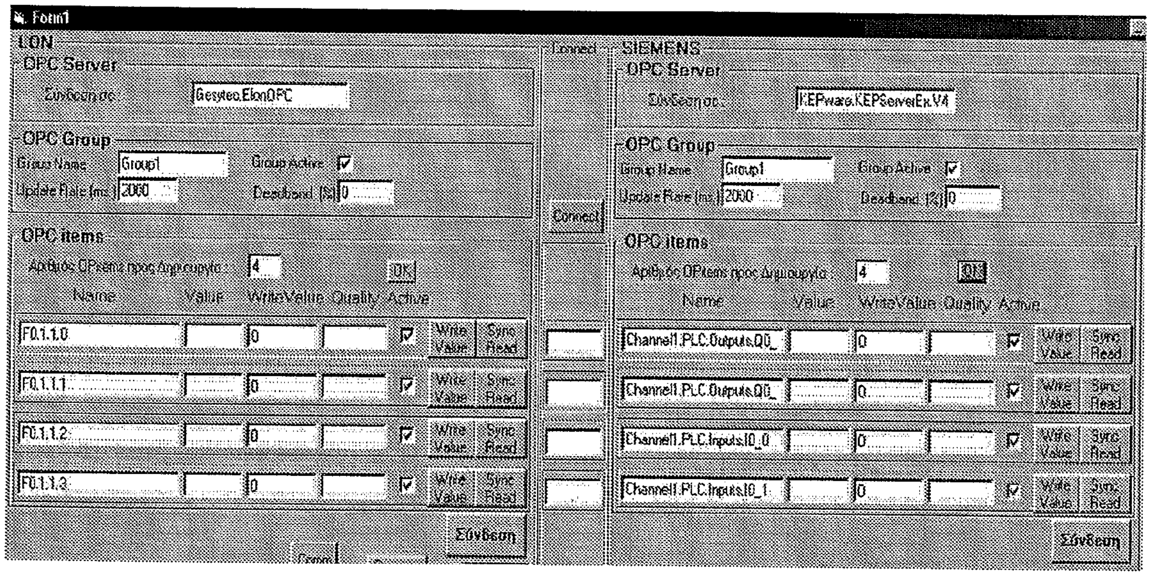

SiemensSw1.1 Object : Siemens
OPCSw1.1Item : OPCItem
SimSw1.1 : Com2StateSwitch
SimSw1.1CTRLSimRel2.1 : ComSwRelCtrlAlg
SimRel2.1 : ComRelay
OPCRel2.1Item : OPCItem
LonRel2.1Object : LonObject
1: Poll Inputs
2: Read SwState
KEPOPC is polling Profibus PLC Inputs
Controller implements the Com2StateSwitch listener
Controller requests ComRelay method Arm() or Disarm()
3: SwitchOpen()
4: Disarm()
5: IsClosed()
6: SwitchClosed()
7: Arm()
VB group item method writes the OPC asynchronously
ELON OPC writes asynchronously the Lonworks NV's
OPC writes the VB group item asynchronously OPC generates the event VB has the listener
8: write new state
9: update state

Fig. 11. A UML sequence diagram of the Sw1.1-to-Rel2.1 control procedure.

schematically in the following UML sequence diagram in Fig. 11.

The real distributed application operates as follows: when the real object Sw1.1 changes state, its value is transferred through Profibus to the IU. KEPWare OPC at the IU Profibus side is periodically polling and reads asynchronously the new Sw1.1 state. Each real sensor or actuator object has a unique OPC address and tag name carrying its current value along with quality and time stamp. VB Profibus side OPCGroup listener listens the Sw1.1 tag data change and stores the new value to the common object "SimSw1.1" corresponding to an OPC specific item. At the same time it provides an input to the "SimSw1.1CTRLSimRel2.1" control algorithm that in turn generates a new output that writes the common object "SimRel2.1" corresponding to another OPC item. Then a VB procedure writes asynchronously the LonWorks side ELON OPC corresponding tag. Finally, ELON OPC writes the corresponding LonWorks NV that controls the real Rel2.1 actuator state. The feedback procedure from LonWorks to Profibus operates similarly. The Visual Basic VA form object with the two OPC servers and four OPC items for each side, linked to the common objects, appears in Fig. 12.

The indicative application operation has been monitored and validated in real time through the Lonbuilder protocol analyzer. This analyzer shows all LonWorks network traffic with packet log, statistics and analysis. All packets exchanged within the LonWorks environment were monitored directly. These packets were generated either by the Profibus node but entered the LonWorks domain through the IU and ELON OPC, or by the LonWorks node. Both OPC servers polling timeouts were programmed to be either short (100–400 ms) or long (10 s). Even if the Profibus side has not been directly monitored, the results of events that occurred there could be traced and indirectly monitored at the LonWorks side. For example, Profibus side input switch state changes resulted in



Form1

LON

OPC Server

Σύνδεση με: GesytecElonOPC

OPC Group

Group Name: Group1 Group Active: ✓

Update Rate (ms): 2000 Deadband (%): 0

OPC items

Αριθμός OPitems προς Δημιουργία: 4

Name	Value	WriteValue	Quality	Active
F0.1.1.0		0		✓
F0.1.1.1		0		✓
F0.1.1.2		0		✓
F0.1.1.3		0		✓

SIEMENS

OPC Server

Σύνδεση με: KEPware.KEPServerEx.V4

OPC Group

Group Name: Group1 Group Active: ✓

Update Rate (ms): 2000 Deadband (%): 0

OPC items

Αριθμός OPitems προς Δημιουργία: 4

Name	Value	WriteValue	Quality	Active
Channel1.PLC.Outputs.Q0_		0		✓
Channel1.PLC.Outputs.Q0_		0		✓
Channel1.PLC.Inputs.I0_0		0		✓
Channel1.PLC.Inputs.I0_1		0		✓

Fig. 12. The Visual Basic VA form object.

packets sent from ELON OPC to LonWorks relay node (Ackd type packets) based on an event driven type of communication. ELON OPC polling of LonWorks node inputs (through NVvalueFetch) resulted in subsequent output changes at the Profibus node. OPC polling timeouts and quantity of OPC items to be polled are definitive for the distributed application closed-loop time constant. This favors rather slow changing applications whose time constant is longer than OPC delays. OPC event driven operation instead of polling can drastically improve the speed of input variables read procedure. VA asynchronous write operations towards the OPC servers and subsequent OPC write operations towards the fieldbus nodes are event driven (concerning ELON), depend on Visual Basic/Windows platform and are faster than polling as well as OPC client–server connection. Exact measurements and a real time operation analysis is beyond the scope of the current work and will be presented at a later time within a real time-oriented interoperability framework.

5. Conclusions

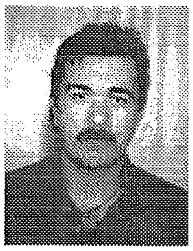
In this paper, an integrated approach is proposed implemented and monitored for the interoperability within a distributed industrial networking environment, which consists of heterogeneous fieldbuses. An interoperability layer is introduced above the OSI-RM application layer, extending the ISA/SP50 User Layer philosophy that provides a common object model and messages domain framework for automation applications. The real distributed application is mapped to a VA within this interoperable domain that links the distributed parts and supports their interoperability. Data-driven messaging is employed within the domain, through supplier common object method calls, performing implicit event transfer. An IU unit was designed, developed and implemented to provide interoperability between a LonWorks and a Profibus fieldbus segments that support a distributed closed-loop automation application.

References

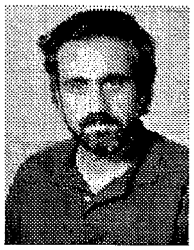
[1] L. Rosen, K. Oberland, The Interoperability Between Real-Time and Non Real-Time Processing on Windows NT Platform, Mitre Technical Report, November 1999.

[2] L. Hadellis, S. Koubias, An Approach to Interoperability in a Heterogeneous Control Network Environment, IFAC Conference on Manufacturing, Modeling, Management and Control (MIM 2000), Patras, Greece, 12–14 July 2000, pp. 98–105.

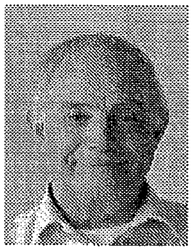
[3] D. Dietrich, T. Sauter, Evolution Potentials for Fieldbus Systems, Vienna University of Technology, IEEE WFCS2000, Porto, Portugal, 6–8 September 2000, pp. 343–350.
[4] E. Schneider, V. Thamburaj, Using LonWorks Controller-Objects in Distributed Nodes, TLON GmbH, LUI, Amsterdam, http://www.tlon.de.
[5] IEC Technical Committee TC65/WG6, IEC61499 Industrial-Process Measurement and Control—Specification IEC Draft 2000.
[6] IEC Committee No. 65C, IEC61804 General Requirements Specification IEC Draft 2000.
[7] Profibus Specification, Normative Parts of PROFIBUS-FMS, -DP, -PA, According to the European Standard EN 50 170, vol. 2, 1.0 ed., March 1998.
[8] Profibus PA Profile for Process Control Devices, Version 3.0, April 1999.
[9] Profinet Architecture Description and Specification Draft, Version V 1.0, August 2001.
[10] LonMark Application Layer Interoperability Guidelines, LonMark Interoperability Association, http://www.lonmark.org.
[11] LonMark Layers 1–6 Interoperability Guidelines, LonMark Interoperability Association, Version 3.0, http://www.lonmark.org.
[12] The Lonpoint System/LonMaker Integration Tool, http://www.echelon.com.
[13] OLE for Process Controls (OPC) Data Access Automation Interface Standard, Version 2.02, OPC Foundation, http://www.opcfoundation.org, 1999.
[14] ISA/IEC SP50 Committee Draft Standard for the User Layer, 1993.
[15] M. Marcos, I. Calvo, D. Orive, I. Sarachaga, J.M. Fuertes, P. Marti, R. Villa, S. Buzoianu, Object-oriented modeling for remote monitoring of manufacturing processes, in: Proceedings of the Eighth IEEE International Conference on Emerging Technologies and Factory Automation, France, 2001, pp. 287–293.
[16] P. Marti, et al., Distributed Supervision and Control of Fieldbus-Based Industrial Processes, IEEE WFCS2000, Porto, Portugal, 6–8 September 2000, pp. 11–18.
[17] PABADIS, Plant Automation Based on Distributed Systems, http://www.pabadis.org.
[18] K. Thramboulidis, C. Tranoris, An architecture for the development of function block oriented engineering support systems, in: IEEE International Symposium on Computational Intelligence in Robotics and Automation, Banff, Alberta, Canada, 2001.
[19] P. Neumann, R. Simon, C. Diedrich, M. Riedl, Field device integration, in: Proceedings of the Eighth IEEE International Conference on Emerging Technologies and Factory Automation, France, 2001, pp. 63–68.
[20] K. Thramboulidis, A. Prayati, Field device specification for the development of function block oriented engineering support systems, in: Proceedings of the Eighth IEEE International Conference on Emerging Technologies and Factory Automation, France, 2001, pp. 581–587.
[21] OPC–DX Data Exchange, http://www.opcfoundation.org, September 2001.



Loukas Hadellis received the diploma in electrical engineering from University of Patras, Greece, in 1981 and the Master of Engineering degree from Carleton University, Ottawa, Canada, in a joint project with Bell-Northern Research in 1983. Worked as research engineer at the University of Patras in EU R&D programs (ESPRIT, RACE) in fiber-optic networks (1984–1987), as production engineer in EGL-Western Greece Paper-Mills S.A. in Industrial Automation Systems (1987–1990) and as a Technical Manager in EU R&D programs (AIM, Telematique) in computer networks at the Institute of Biomedical Technology–INBIT (1991–1995). Since 1995, he is assistant professor at the Technological Educational Institute of Patras and Technical Manager of the Network Operation Center (NOC) and Delaurs 2 EU Gunet/Grnet R&D Institute programs. His current research interests include networked microcontrollers, fieldbuses, industrial informatics and building/home automation networking systems. He is member of the Hellenic Technical Chamber (TEE), the Hellenic Association of Dipl. Mechanical & Electrical Engineers and the Hellenic Association of Technical Scientists in Industry.



Prof. S. Koubias received the diploma in electrical engineering and the PhD degree from the University of Patras, Greece, in 1976 and 1982, respectively. Since 1999, he has been an associate professor in the Department of Electrical Engineering of the University of Patras, Greece. From 1976 to 1999, he was a research associate, lecturer, assistant and associated professor at the Department of Electrical Engineering, University of Patras, Greece. From March 2001, he is the Director of the Network Operation Center (NOC) of the University of Patras. From June 1998, he is member of the Industrial Systems Institute (ISI) of Patras. His current research interests include real time communication protocols, industrial networking systems, wireless networks, multimedia communications, embedded systems, advanced microprocessor architectures and home networking systems. He has participated as co-ordinator, partner in many Greek and European R&D programs. He has published over 90 papers in miscellaneous international journals and conference proceedings. He has also written books and course notes in industrial networks, microprocessors and microcontrollers. Prof. Koubias is a member of IEEE and Technical Chamber of Greece (TEE).



Vassilios Makios was born in Kavala, Greece. Received his Electrical Engineering degree (Dipl. Ing.) from Technical University in Munich, Germany in 1962 and PhD degree (Dr. Ing.) from Max Planck Institute for Plasmaphysics and Technical University in Munich in 1966. From 1962 to 1967 he was research associate in the Max Planck Institute for Plasmaphysics in Munich, in microwave interaction studies of plasmas. Served as assistant professor (1967–1970), associate professor (1970–1973) and full professor (1973–1977) in Department of Electronics, Carleton University Ottawa, Canada, involved with teaching-research in microwave-optical communications, radar technology, remote sensing and CO2 lasers. From 1977 he is honorary research professor of Carleton University. Since 1976 he is professor of Engineering and Director of Electromagnetics Laboratory in Electrical Engineering Department of University of Patras Greece, involved in teaching-research in microwave-optical communications, data communications networks, LANs, MANs, B-ISDN, ATM. Published over 140 papers, holds numerous patents in the above fields. Participated in organizing committees of numerous IEEE and European Conferences, technical program chairman of 5th Photovoltaic European Community Conference in Athens 1983 and co-chairman of European Community EURINFO 1988 Conference. Received silver medal (1984) and golden medal (1999) of German Electrical Engineering Society (VDE) and IEEE 2000 Chester Sall Award for 2nd place Transaction Award Paper. Senior member of IEEE, member of Canadian Association of Physicist, German Physical Society and VDE, professional engineer of Province of Ontario and member of Greek Technical Chamber.